Efficient Estimation of Stochastic Flow Network Reliability

Héctor Cancela
Universidad de la República
Montevideo, Uruguay
cancela@fing.edu.uy

Leslie Murray
FCEIA–Universidad Nacional de Rosario
Rosario. Argentina
leslie@eie.fceia.unr.edu.ar

Gerardo Rubino
INRIA, univ. Rennes, CNRS, Irisa, France
Rennes, France
Gerardo.Rubino@inria.fr

***Abstract*—The Creation Process is an algorithm that transforms a static network model into a dynamic one. It is the basis of different variance reduction methods designed to make efficient reliability estimations on highly reliable networks in which links can only assume two possible values, operational or failed. In this article the Creation Process is extended to let it operate on network models in which links can assume more than two values. The proposed algorithm, called here Multi–Level Creation Process, is the basis of a method, also introduced here, to make efficient reliability estimations of highly reliable stochastic flow networks. The method proposed, which consists in an application of Splitting over the Multi–Level Creation Process, is empirically shown to be accurate, efficient, and robust.**

***Index Terms*—stochastic flow network, capacity, reliability, estimation.**

ACRONYMS[1]

MLCP	Multi–Level Creation Process
r.v.	random variable
i.i.d.	independent and identically distributed
s.t.	such that
i.e.	that is
w.p.	with probability

NOTATION

M	maximum amount of flow generated in node $\mathbf{s}$ that can reach node $\mathbf{t}$
T	value such that the *reliability* (*unreliability*) of a *stochastic flow network* is the probability that M is above (below) T
n	number of nodes in the network
m	number of links in the network
$\mathcal{V}$	set of n nodes
$\mathcal{E}$	set of m links
X_l	random capacity of link l
x_l	a realization of the r.v. X_l
Ω_l	state space for r.v. X_l
$\mathbf{X}$	random vector $(X_1, X_2, \ldots, X_m)$ over state space $\mathbf{\Omega}$ indicating the links' capacities
$\mathbf{x}$	a realization of the random vector $\mathbf{X}$
$\mathbf{\Omega}$	m–dimensional state space $(\Omega_1 \times \ldots \times \Omega_m)$ for the random vector $\mathbf{X}$
N	sample size of a Monte Carlo experiment
$\mathbf{X}^{(j)}$	sample of the random vector $\mathbf{X}$, $j = 1, \ldots, N$
$M(\mathbf{X})$	maximum amount of flow generated in node $\mathbf{s}$ that can reach node $\mathbf{t}$ as a function of vector $\mathbf{X}$.
$\mathcal{G}$	graph $(\mathcal{V}, \mathcal{E}, \mathbf{X})$ representing a stochastic flow network topology
$f_{\mathbf{X}}(\mathbf{x})$	probability distribution of r.v. $\mathbf{X}$
p_i	probability of the i^{th} outcome in a probability mass function
$\widehat{p_i}$	unbiased estimator of p_i
ζ	$\mathbb{P}\{M(\mathbf{X}) < T\}$: *unreliability* of a *stochastic flow network*
Q	*unreliability* of a *basic connectivity network model*
$\widehat{\zeta}$	unbiased estimator of ζ
$\widehat{\zeta}_c$	crude or standard estimator of ζ
M_l	maximum capacity of link l for the case of only two possible values, 0 or M_l
$M_{l,k}$	capacity of link l for the case of n_l capacity values: $0 < M_{l,1} < M_{l,2} < \ldots < M_{l,n_l}$
$[M_n, M_1]$	interval of possible non–zero capacity values when capacity is uniformly distributed
M_{n+1}	auxiliary bound s.t. when the reparation times are sampled beyond $t = 1$, the "fictitious" capacities associated are in (M_{n+1}, M_n)

[1]The singular and plural of an acronym are always spelled the same.

$\phi(\mathbf{X})$	indicator function of event $\{M(\mathbf{X}) < T\}$
r_l	*reliability* of link l
q_l	*unreliability* of link l: $1 - r_l$
τ_l	reparation time of link l
λ_l	rate of the exponentially distributed repair time τ_l
$X_l(t)$	capacity evolution of link l: $X_l(t) = 0$ if $t < \tau_l$, $X_l(t) = 1$ if $t \geq \tau_l$
$\mathbf{X}(t)$	$(X_1(t), X_2(t), \ldots, X_m(t))$
$M(\mathbf{X}(t))$	same as $M(\mathbf{X})$ when $\mathbf{X}$ becomes the random process $\mathbf{X}(t)$
τ_c	reparation time of the *critical link*: e_c
e_c	link upon whose reparation the flow $M(\mathbf{X}(t))$ reaches the value T
$G(\tau_c)$	indicator function of event $\{\tau_c > 1\}$
t_j	temporal bound s.t. $0 \leq \tau_l < t_1$ w.p. p_1, …, $t_{n-1} \leq \tau_l < t_n$ w.p. p_n, and $\tau_l > t_n$ w.p. p_0, $j = 1, \ldots, n$
t_{ij}	same as t_i when there is one different temporal bounds set for each link i
$X(t)$	Markov process with discrete state space $\mathcal{X}$
$\mathcal{X}_A, \mathcal{X}_B$	two disjoint regions in $\mathcal{X}$
κ_A, κ_B	instants when $X(t)$ enters, resp., $\mathcal{X}_A$ and $\mathcal{X}_B$
$h(x)$	importance function, $h : \mathcal{X} \to \mathbb{R}$
q	number of thresholds or bounds in Splitting
ℓ_k	one of the thresholds on the domain of $h(x)$: $\ell_0 = 0 < \ell_1 < \ell_2 < \cdots < \ell_q = \ell$
κ_0	$\inf\{t > 0 : h(X(t)) = \ell_0\}$
κ_k	$\inf\{t > 0 : h(X(t)) = \ell_k > h(X(t^-))\},\ k \geq 1$
D_k	event $\{\kappa_k < \kappa_0\},\ k = 1, 2, \cdots, q,$
N_k	number of Splitting trajectories started from threshold k, $k = 0, \ldots, q-1$
R_k	number of Splitting trajectories that reach threshold k, $k = 1, \ldots, q$
α_k	ratio between *in* and *out* trajectories per level, $N_k/F_k,\ k = 1, \ldots, q-1$, in *Fixed Spliting*
F_k	$N_k,\ k = 0, \ldots, q-1$ total number of out trajectories per level in *Fixed Effort*
Γ	$\{\tau_{(1)}, \cdots, \tau_{(m)}\}$: *order statistics* of the set of repair times $\{\tau_1, \cdots, \tau_m\}$
$\tau_{(i)}$	i^{th} element in Γ
Γ^i	a partially sampled sequence $\{\tau_{(1)}, \tau_{(2)}, \cdots, \tau_{(i)}\}$
Λ^i	set of parameters λ of the links whose repair times are in Γ^i
S^i	sum of the rates in Λ^i: $\sum_{j:\lambda_j \in \Lambda_i} \lambda_j$
$\bar{S}^i$	sum of the rates not in Λ^i: $\sum_{j:\lambda_j \notin \Lambda_i} \lambda_j$,
Δ	exponentially distributed random variable with parameter $\bar{S}^i$
u_k	one of the thresholds on the domain of $M(\mathbf{X}(t))$: $u_0 = 0 < u_1 < u_2 < \cdots < u_q = 1$
E	event $\{M(\mathbf{X}(1)) < T\}$
E_k	event $\{M(\mathbf{X}(u_k)) < T\},\ k = 1, 2, \cdots, q$
$\mathbb{RE}$	*Relative Error*: $\mathbb{V}\{\widehat{\zeta}\}^{1/2}/\mathbb{E}\{\widehat{\zeta}\}$
$\mathbb{W}$	*Precision Gain*: $(\mathbb{V}\{\widehat{\zeta_c}\} \times t_c)/(\mathbb{V}\{\widehat{\zeta}\} \times t)$
$\mathcal{G}_R$	*residual graph* $(\mathcal{V}, \mathcal{E}, \mathbf{X}_R)$
$\mathbf{X}_R$	random vector $(X_{1R}, X_{2R}, \ldots, X_{mR})$ indicating the links' *residual capacities*
Y_i	net flow passing through link l_i

I. Introduction

This article introduces a Monte Carlo method for estimating efficiently the *reliability* of highly reliable *stochastic flow networks*. While some exact methods for computing this reliability index have been published (see for instance [1] and [2]), this is an NP-hard problem and exact computation can only be applied to small sized networks. Standard simulation techniques do not suffer from this scalability issue as far as system's reliability is not too close to one, but they can not achieve efficient estimations on highly reliable systems' *reliability* because the high number of replications required to simulate system's failures causes excessively long simulation times. One possible solution to this problem is to design estimators with lower variance than the variance of the standard one. In the case of *stochastic flow networks*, variance reduction methods have been proposed within three research lines based, respectively, in *graph evolution models*, *Generalized Splitting* and *bounds* [3]. The methods based on *graph evolution models* [4] transform the static model into a dynamic one by means of an artificial time and make use of techniques like Permutation Monte Carlo and C–Spectrum. *Generalized Splitting* is an adaptation to static models of the classical Splitting technique. It has been studied for the *stochastic flow network* problem by Botev et al. [5]. The methods based on *bounds* derive from a well–known technique due to George S. Fishman, which was widely applied in the context of Monte Carlo simulation and it is successfully adapted to *stochastic flow networks* in [6] and [7].

In a *stochastic flow network* there is one source node, say $\mathbf{s}$, where a certain amount of flow is generated, and one target (or sink) node, say $\mathbf{t}$, where all the flow coming from $\mathbf{s}$ is consumed. In every node the incoming flow is equal to the outgoing flow, except for $\mathbf{s}$, where the outgoing flow is higher than the incoming flow and for $\mathbf{t}$, where the incoming flow is higher than the outgoing flow. The maximum amount of flow that a link is able to transport is called its *capacity*. For any given set of links' *capacities*, the maximum amount of flow generated in $\mathbf{s}$ that can reach $\mathbf{t}$ is denoted by M. When the links fail, their *capacities* fall randomly (down to one of several possible values). This means that, in the network model, the individual links' *capacities* and the value of M are all random variables. The most common measure of *reliability* (*unreliability*) for *stochastic flow networks* is the probability that M is above (below) a given threshold T.

One of the approaches to estimate this probability is based on *graph evolution models* [4]. This method depends on the introduction of an artificial time to transform the original static model into a dynamic one. Based on this transformation, the accuracy of the simulation estimations is improved. Basically, the system is assumed to start at $t = 0$ with all its components *failed* (*repaired*); afterwards, each component becomes *repaired* (*failed*) after a random time whose distribution parameters are related to its own *reliability* (*unreliability*). The time when the whole system becomes *repaired* (*failed*) is a random variable, whose expectation is directly related to the

system *reliability* (*unreliability*). It is a well–known technique mostly applied on the *basic connectivity network model* in which links can only be found in one of two possible states.

Another approach [5] applies the Splitting technique to trajectories that simulate the network evolution as a Markov stochastic process through its state space. Rare events correspond to regions of the state space that are reached with very low probability. Splitting works by partitioning the state space into nested subspaces, where the event of interest is the innermost one, and by splitting (multiplying) recursively trajectories when they enter subspaces that are closer to the rare event. When correctly weighted, the estimations that arise from splitting trajectories can be used to obtain an unbiased estimator of the reliability of the original system. Proceeding this way, the most promising trajectories are privileged, leading to a reduced estimator variance. Generalized Splitting is a variant of Splitting thought of as to be applied on static models as well as dynamic ones. In a static model like a *stochastic flow network* there is no temporal evolution; this is the reason why it is not possible to follow trajectories and, for instance, to detect the crossing points through the subspaces borders. To address this issue, Generalized Splitting introduces auxiliary random variables conditioned on events representing the evolution through the different subspaces. The rest, i.e. the mechanism of recursively splitting (multiplying) the simulation, re–starting from certain points, remains the same.

Two other methods proposed in the field of *stochastic flow networks*, [7] and [6], are based on changing the probability distributions of the individual links' *capacities*. The objective is that under the new probability distributions the *reliability* estimator is still unbiased with respect to the real value, but has less variance than the standard one. In [7] and [6] the change of the probability distributions is based on *bounds* that come from a decomposition of the state space due to Doulliez and Jamoulle [8].

The method proposed in this article is a generalization of the one introduced in [9], where the model under analysis was not a *stochastic flow network* but a *basic connectivity network model* for which the *reliability* is the probability that two selected nodes are connected by a path formed by *operational* links, provided that each link can be either *failed* or *operational*. In [9] Splitting is applied to the Creation Process [10], a graph evolution model in which all the links start *failed* and become *operational* in random exponentially distributed times. The Creation Process applies to models for which the components can only be found in one of two states. In this article (where, as stated, links' *capacities* can be assume one of many possible levels) the Creation Process is transformed into what we called Multi–Level Creation Process (MLCP), in order to simulate the links' temporal evolution of a *stochastic flow network*, and to let Splitting be applied exactly as it was in [9].

The subsequent sections of this article address the following contents: for Section II, an introduction of the network model; for Section III, the MLCP presentation and for Section IV, the basis of Splitting. Section V shows the application of Splitting on the MLCP, Section VI gives some implementation guidelines and Section VII presents some experimental results. Section VIII discusses the future lines of work and outlines the final conclusions.

II. Network Modeling

A *stochastic flow network* can be be represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is a set of n nodes, $\mathcal{E}$ a set of m links, and $\mathbf{X} = (X_1, \ldots, X_m)$ a *capacity* random vector over the state space $\mathbf{\Omega} = (\Omega_1 \times \ldots \times \Omega_m)$. Here, Ω_i denotes the space of possible values for the *capacity* of link i, $i = 1, \ldots, m$. The distribution $f_{\mathbf{X}}(\mathbf{x})$ of $\mathbf{X}$ can be either continuous or discrete. In the latter case f is a probability mass function: $f_{\mathbf{X}}(\mathbf{x}) = \mathbb{P}\{\mathbf{X} = \mathbf{x}\},\ \forall \mathbf{x} \in \mathbf{\Omega}$.

When $M(\mathbf{X})$ denotes the random variable that indicates the maximum amount of flow with origin in $\mathbf{s}$ that can reach $\mathbf{t}$ for a certain vector of links' *capacities*, $\mathbf{X}$, the *unreliability* of the *stochastic flow network* is defined as $\zeta = \mathbb{P}\{M(\mathbf{X}) < T\}$. If the network is highly reliable, we have $\zeta \ll 1$.

Given a value $\mathbf{x}$ of the random vector $\mathbf{X}$, it is possible to compute $M(\mathbf{x})$ employing one of several existing algorithms, like Ford–Fulkerson's [11], whose computing time complexity is $O(mC)$, where C is the largest *capacity* in the network, Edmonds–Karp's [12], whose computing time complexity is $O(nm^2)$, or Goldberg–Rao's [13] with time complexity $O(\min(n^{2/3}, m^{1/2})\, m\, \log(n^2/m)\, \log(C))$.

It is reasonable to assume that every link has a finite maximum *capacity*, due to physical limitations. This means that, in most cases, there should be an upper bound on the components' values of $\mathbf{X}$. It is also reasonable to accept that in the event of a total failure, a link may not be able to transport any amount of flow. This condition sets the minimum possible flow per link to zero, but not below, which means that the components of $\mathbf{X}$ could never be negative. According to these constraints, the simplest form for $\mathbf{X}$ is the one proposed in [4], where every X_i is a discrete random variable over $\Omega_i = \{0, M_i\},\ M_i > 0,\ i = 1, \ldots, m$. Links can be in one of two possible conditions, fully *operational*, i.e., able to transport flow up to the value M_i, or absolutely *failed* and not able to transport any flow. A broader range of failure values is offered by the state space used in [7] and [6], where $\Omega_i = \{0, M_{i,1}, M_{i,2}, \ldots, M_{i,n_i}\}$, with $0 < M_{i,1} < M_{i,2} < \ldots < M_{i,n_i} < +\infty,\ i = 1, \ldots, m$. A generalization of the former ones is a continuous distribution on the state space $\mathbf{\Omega} = [0, +\infty)^m$. This distribution was suggested in [5], although the specific distribution used in that article for the experimental tests was continuous on $\Omega_i = [0, M_i),\ M_i > 0,\ i = 1, \ldots, m$.

Given the following indicator function:

$$\phi(\mathbf{X}) = \begin{cases} 1 & \text{if } M(\mathbf{X}) < T \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

if the components of $\mathbf{X}$ are discrete random variables, the *unreliability* can be written as:

$$\zeta = \mathbb{P}\{\phi(\mathbf{X}) = 1\} = \mathbb{E}\{\phi(\mathbf{X})\} = \sum_{\mathbf{x} \in \mathbf{\Omega}} \phi(\mathbf{x})\, \mathbb{P}\{\mathbf{X} = \mathbf{x}\}, \tag{2}$$

whereas, if the components of $\mathbf{X}$ are continuous:

$$\zeta = \int_{\mathbf{x}\in\mathbf{\Omega}} \phi(\mathbf{x})\, f_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x}. \tag{3}$$

Exact computation of ζ belongs to the NP–hard complexity class [14]. Computing ζ for large or even medium–sized networks is computationally intractable. Monte Carlo estimations provide an alternative solution. The crude or standard Monte Carlo is the most straightforward technique in this family. A standard Monte Carlo estimation of ζ is given by:

$$\widehat{\zeta} = \frac{1}{N}\sum_{j=1}^{N} \phi(\mathbf{X}^{(j)}), \tag{4}$$

where $\mathbf{X}^{(j)},\ j = 1,\ldots,N$, is a set of N i.i.d. values sampled from the distribution of $\mathbf{X}$. When the states of the links are mutually independent, the samples can be obtained component–wise from the distributions of each X_i, $\forall X_i \in \Omega_i,\ i = 1,\ldots,m$. The value $\widehat{\zeta}$ is an unbiased point estimator of ζ.

A typical measure of accuracy for this estimator is the relative error defined as $\mathbb{V}\{\widehat{\zeta}\}^{1/2}/\mathbb{E}\{\widehat{\zeta}\} = \mathbb{V}\{\widehat{\zeta}\}^{1/2}/\zeta$, where $\mathbb{V}\{\widehat{\zeta}\} = \zeta\,(1-\zeta)/N$ is the variance of $\widehat{\zeta}$. If, for any fixed value of N, ζ tends to 0 —indicating that the network becomes more reliable— the relative error grows boundlessly. This is an important limitation of crude Monte Carlo. In order to ensure an acceptable accuracy level it is necessary to find other variants for the highly reliable case, in which the standard deviation, ζ, goes to zero faster than $\mathbb{V}\{\widehat{\zeta}\}^{1/2}$. The so–called *variance reduction* techniques, the application of which has led to the design of different *reliability* estimators in the context of *stochastic flow networks*, try to achieve this goal.

III. Multi–Level Creation Process

In the *stochastic flow network* model proposed by Gertsbakh et al. in [4], every X_i in $\mathbf{X}$ is an independent discrete random variable over $\Omega_i = \{0, M_i\},\ M_i > 0,\ i = 1,\ldots,m$. There, $X_i = 0$ means that the i^{th} link is *failed* and, therefore, it is not able to transport any flow, whereas $X_i = M_i$ indicates that the i^{th} link is fully *operational* and able to transport flow up to its *capacity* M_i. The number $\mathbb{P}\{X_i = 0\} = q_i$ is the individual link i's *unreliability*, whereas $\mathbb{P}\{X_i = M_i\} = 1 - q_i = r_i$ is its *reliability*.

Based on this model, it is possible to develop a Creation Process corresponding to a *stochastic flow network* using a similar approach as for the case of a *classical connectivity network model*. With this purpose, the state vector $\mathbf{X}$ is transformed into a random process $\{\mathbf{X}(t), t \geq 0\}$ that operates in the following way: at time $t = 0$ all the links are *failed* ($X_i(0) = 0,\ \forall i$) and they become *operational* (repaired) at times $\tau_i,\ i = 1,\cdots,m$, exponentially distributed, with rates $\lambda_i = -\ln q_i$. The t index is an artificial time, our initial model is static. According to their distributions, the probability that link e_i is repaired before or at time $t = 1$ is $\mathbb{P}\{\tau_i \leq 1\} = 1 - e^{-\lambda_i} = 1 - q_i = r_i$, whereas the probability that link e_i is repaired beyond $t = 1$ is $\mathbb{P}\{\tau_i > 1\} = q_i$.

Figure 1 (b) shows a sample $\tau_i < 1$, with the process $X_i(t)$ changing from 0 to M_i at time $t = \tau_i$ and remaining in M_i forever. At $t = 1$ link i is already repaired with probability $r_i = 1 - q_i$, and it is not yet repaired with probability q_i (that is, $\mathbb{P}\{X_i(1) = 0\} = q_i$). Therefore, to see a snapshot of the dynamic system at $t = 1$ is equivalent to observing the static system. Thus, at $t = 1$ the maximum possible amount of flow originated at $\mathbf{s}$ and consumed at $\mathbf{t}$ is at most T with probability ζ, that is $\mathbb{P}\{M(\mathbf{X}(1)) < T\} = \zeta$.

If there exists at least one vector $\mathbf{x}$ for which $M(\mathbf{x}) \geq T$ (i.e., T is a reachable flow value in this network), then $M(\mathbf{X}(t))$ will reach —or exceed— T at a time $t = \tau_c$ which is the repair time of a link e_c, denoted as the *critical link* (observe that, as a consequence, the critical link is a random object). As ζ is the probability that $M(\mathbf{X}(t))$ is less than T at time $t = 1$, ζ can also be defined as the probability that $M(\mathbf{X}(t))$ reaches —or exceeds— T later than $t = 1$, that is:

$$\zeta = \mathbb{P}\left\{\tau_c > 1\right\}. \tag{5}$$

If the indicator function of the event $\{\tau_c > 1\}$ is denoted by $G(\tau_c)$,

$$G(\tau_c) = \begin{cases} 1 & \text{if } \tau_c > 1 \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

then ζ can be computed as the expected value of $G(\tau_c)$:

$$\zeta = \mathbb{E}\left\{G(\tau_c)\right\}. \tag{7}$$

We can denote $\widehat{\zeta_c}$ the crude or standard estimation of ζ obtained by means of this formula:

$$\widehat{\zeta_c} = \frac{1}{N}\sum_{n=1}^{N} G(\tau_c^{(n)}). \tag{8}$$

This determination consists in sampling independently N times the network repair time, $\tau_c^{(n)}\ n = 1,\cdots,N$, and to compute the proportion of them for which $\tau_c^{(n)} > 1$.

The random process introduced so far is the Creation Process. It may apply for a standard estimation, as shown in (8), but it can also be the basis of more accurate estimation methods. One of them is Permutation Monte Carlo [10], [4], a technique where the event $\{\tau_c > 1\}$ is conditioned on the order in which the links are repaired, provided that for every set of links' repairing order the probability $\mathbb{P}\{\tau_c > 1\}$ can be computed exactly. Then, after sampling independently N times the repairing order of all the links, the final estimation of ζ is the average of the probabilities $\mathbb{P}\{\tau_c > 1\}$. Another of the methods based on the Creation Process is Splitting/CP [9], a technique in which the sequences of repair times are seen as trajectories through the state space, and they are multiplied (cloned) as they cross thresholds on their way to the target event that is $\{\tau_c > 1\}$ (this method will be addressed again in Sections IV and V).

The MLCP, which is the core of this article's proposal, will be now introduced with the support of Figure 2. In a first approach and without any loss of generality, let us consider a *stochastic flow network* model in an homogeneous setting in which $\Omega_1 = \cdots = \Omega_m = \{M_1,\ldots,M_n,0\}$, with $\infty > M_1 > \cdots > M_n > 0$, where $X_i = M_j$ with probability $p_j,\ j = 1,\ldots,n$, and $X_i = 0$ with probability p_0.

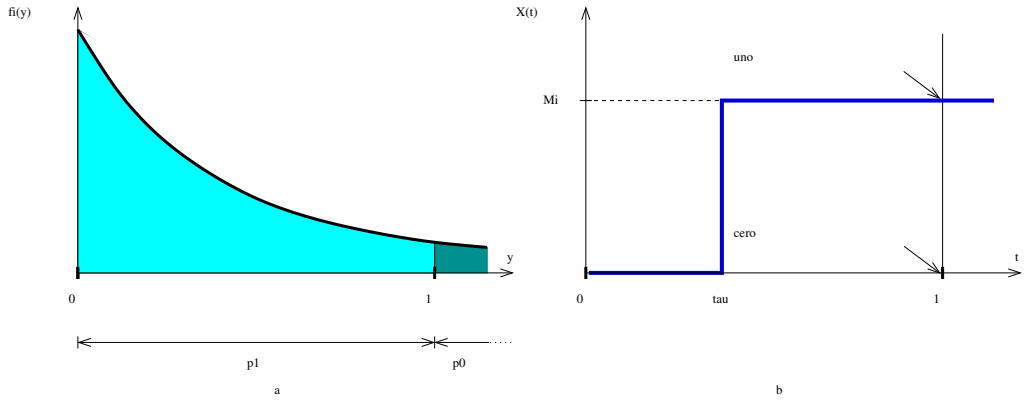

Fig. 1: The Creation Process

As in the Creation Process, let τ_i be the repair time of the i^{th} link, $i = 1, \ldots, m$, and let $\{t_1, t_2, \ldots, t_n\}$, $0 < t_1 < \ldots < t_n = 1$, be a partition on $[0, \infty)$, such that times, τ_i, fall into the intervals determined by the partition, according to the following rules: $0 \leq \tau_i < t_1$ with probability $p_1, \ldots, t_{n-1} \leq \tau_i < t_n$ with probability p_n, and $\tau_i > t_n = 1$ with probability p_0. Event $t_{k-1} \leq \tau_i < t_k$ has the same probability as event $X_i = M_k$, $k = 1, \ldots, n$, whereas event $\tau_i > 1$ has the same probability as event $X_i = 0$. Thus, the value of X_i can be sampled implicitly, according to the interval in which the repair time, τ_i, falls, what, in the end, means that the value of X_i is a function of the time τ_i, as it was in the original Creation Process.

It is necessary now to determine the rate of the exponentially distributed repair times (that for simplicity in this example, and without any loss of generality, are proposed to be all equal, $\lambda_i = \lambda$, $i = 1, \ldots, m$) and the times t_1, t_2, ..., t_{n-1}, in order to guarantee that, at $t = 1$, the *capacity* of every link will take a value M_j with probability p_j, $j = 1, \ldots, n$, or 0 with probability p_0. If this is achieved, at $t = 1$ the maximum amount of flow generated in $\mathbf{s}$ that can reach $\mathbf{t}$ will be below T, with probability ζ.

The determination of λ is straightforward:

$$\mathbb{P}\{\tau_i > 1\} = 1 - F_{\tau_i}(1) = e^{-\lambda} = p_0, \tag{9}$$

then,

$$\lambda_i = \lambda = -\ln(p_0), \ i = 1, \ldots, m. \tag{10}$$

The rest of the parameters are obtained as follows:

$$\mathbb{P}\{\tau_i < t_1\} = F_{\tau_i}(t_1) \tag{11}$$

$$= 1 - e^{-\lambda t_1} = p_1 \quad \Rightarrow \quad e^{-\lambda t_1} = 1 - p_1, \tag{12}$$

$$\mathbb{P}\{t_1 < \tau_i \leq t_2\} = F_{\tau_i}(t_2) - F_{\tau_i}(t_1) \tag{13}$$

$$= (1 - e^{-\lambda t_2}) - (1 - e^{-\lambda t_1}) \tag{14}$$

$$= e^{-\lambda t_1} - e^{-\lambda t_2} \tag{15}$$

$$= (1 - p_1) - e^{-\lambda t_2} = p_2, \tag{16}$$

therefore,

$$e^{-\lambda t_2} = 1 - p_1 - p_2. \tag{17}$$

Clearly: $e^{-\lambda t_j} = 1 - p_1 - \cdots - p_j$, $j = 1, \cdots, n$. As the value of λ is already known, it follows that:

$$t_j = \frac{\ln(1 - p_1 - \cdots - p_j)}{\ln(p_0)}, \ j = 1, \cdots, n. \tag{18}$$

However, in the most general case, the links' individual parameters are not necessarily equal; instead, $\Omega_i = \{M_{i1}, M_{i2}, \ldots, M_{in_i}, 0\}$, $+\infty > M_{i1} > M_{i2} > \ldots > M_{in_i} > 0$, where $X_i(t) = M_{i1}$ with probability $p_{i1}, \ldots, X_i(t) = M_{in_i}$ with probability p_{in_i} and $X_i(t) = 0$ with probability p_{i0}, $i = 1, \ldots, m$, where n_i is the possible number of levels of the i^{th} link. Anyway, the computation of the parameters λ_i, t_{i1}, t_{i2}, ... and t_{in-1}, by means of expressions (10) and (18), is straightforward.

In the analysis introduced so far, the values of $X_i(t)$ belong to a discrete state space and they are assigned according to the exponentially distributed repair time τ_i. Let us now consider a model in which the links are *failed*, and therefore not able to transport any flow, with probability p_0, but whenever they are *operational* their *capacity* values are not discrete random variables, but continuous random variables, instead. The definition of Multi–Level Creation Process still holds, but now X_i equals 0 with probability p_0 and it takes a real value in $[M_n, M_1]$, $0 < M_n < M_1 < +\infty$, with probability $1 - p_0$.

It will now be considered the particular case in which, $X_i = 0$, $i = 1, \ldots, m$, with probability p_0, and $X_i \sim \text{Unif}\,[M_n, M_1]$ with probability $1 - p_0$. Now, uniformly distributed values (*capacities*) must be obtained as a function of the exponentially distributed samples (repair times). Thus, whenever an exponentially distributed time τ_i is sampled in $[0, 1]$, a value in $[M_n, M_1]$ must be computed as a function of this time. And the function that computes these values must guarantee that they are uniformly distributed in $[M_n, M_1]$. It is useful to consider also that whenever the exponentially distributed times τ_i are sampled beyond $t = 1$, the function will still compute uniformly distributed values. However the values assigned to times in $(1, \infty)$ will be in (M_{n+1}, M_n), where $M_{n+1} \leq M_n$, being M_{n+1} an auxiliary value (that could even be negative). Finally, the function that transforms exponentially distributed values into uniformly distributed values has (i) to produce a value in $[M_n, M_1]$ if the sampled time is in $[0, 1]$ and (ii) a value in (M_{n+1}, M_n) if the sampled time is in $(1, \infty)$. In

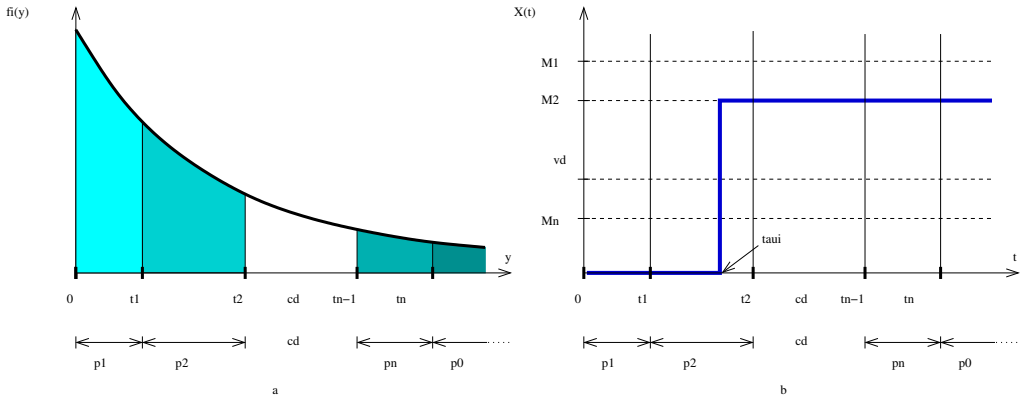

Fig. 2: The Multi–Level Creation Process

the end, the probability of producing a value in $[M_n, M_1]$ will be $1 - p_0$, whereas the probability of producing a value in (M_{n+1}, M_n) will be p_0. The mechanism is illustrated in Figure 3.

As the probability p_0 is known, it can be used to determine M_{n+1} as follows:

$$p_0 = \frac{M_n - M_{n+1}}{M_1 - M_{n+1}} \quad \rightarrow \quad M_{n+1} = \frac{M_n - p_0 M_1}{1 - p_0}. \tag{19}$$

The determination of λ for the i^{th} link is the same as in the discrete model:

$$\mathbb{P}\{\tau_i > 1\} = 1 - F_{\tau_i}(1) = e^{-\lambda} \; = \; p_0, \tag{20}$$

then,

$$\lambda = -\ln(p_0), \; i = 1, \ldots, m. \tag{21}$$

Once the values of M_{n+1} and λ have been computed, it is possible to derive the function that returns uniformly distributed *capacity* values, given exponentially distributed repair times. Figure 4 (a) shows the domains correspondence; for any sampled repair time τ_i, a *capacity* value x_i must be assigned to the variable X_i. Clearly, $\mathbb{P}\{0 \leq \tau_i < 1\} = \mathbb{P}\{M < x_i \leq M_1\}$, where the probability distribution of the term on the left is exponential, and the probability distribution of the term on the right is uniform. The correspondence still holds between certain parts of the referred domains, as shown in Figure 4 (b). In this sense, for example, repair times sampled in $[0, \tau]$ will have associated *capacities* in $[x, M_1]$. Exploring the relation between x and τ will lead to the assignment mechanism that we are looking for:

$$\begin{array}{cc} \text{EXPONENTIAL} & \text{UNIFORM} \\ \downarrow & \downarrow \end{array}$$

$$\mathbb{P}\{0 \leq \tau_i < \tau\} = \mathbb{P}\{x < x_i \leq M_1\} \tag{22}$$

$$1 - e^{-\lambda\tau} = \frac{M_1 - x}{M_1 - M_{n+1}} \tag{23}$$

$$e^{-\lambda\tau} = \frac{x - M_{n+1}}{M_1 - M_{n+1}} \tag{24}$$

$$x = M_{n+1} + e^{-\lambda\tau}\,(M_1 - M_{n+1}) \tag{25}$$

$$x = \frac{M_n - p_0 M_1}{1 - p_0} + e^{-\lambda\tau}\,\frac{M_1 - M_n}{1 - p_0}. \tag{26}$$

Finally, given p_0, M_1 and M_n, the rate for sampling exponentially distributed repair times, τ_i, is $\lambda = -\ln(p_0)$ and, for every sampled τ_i, the associated *capacity* value is $x_i = (M_n - p_0 M_1)/(1 - p_0) + e^{-\lambda\tau_i}\,(M_1 - M_n)/(1 - p_0)$.

In the most general case the links' individual parameters are not necessarily equal; instead, the probabilities of finding a link failed are p_{i0}, $i = 1, \ldots, m$, whereas the *capacity* extremes are M_{i1} and M_{in}. Anyway, for the most general case, the method operates as well and the parameters' computation by means of expressions (21) and (26) is straightforward.

IV. Splitting

Let $\{X(t),\ t \geq 0\}$ be a Markov process with discrete state space $\mathcal{X}$, and let $\mathcal{X}_A$ and $\mathcal{X}_B$ be two disjoint regions in $\mathcal{X}$. Assume $X(0) \notin \mathcal{X}_B$. A quantity that very frequently is of interest in different performance and dependability problems is the probability ζ that, starting at $t = 0$, process $X(t)$ enters $\mathcal{X}_B$ without having entered $\mathcal{X}_A$ before. If κ_A is the instant when $X(t)$ enters $\mathcal{X}_A$ the first time (or comes back to $\mathcal{X}_A$ if $X(0) \in \mathcal{X}_A$) and κ_B is the instant when $X(t)$ enters $\mathcal{X}_B$ the first time, then $\zeta = \mathbb{P}\{\kappa_B < \kappa_A\}$. Regions $\mathcal{X}_A$ and $\mathcal{X}_B$ may be defined implicitly via an importance function $h : \mathcal{X} \rightarrow \mathbb{R}$ as follows: $\mathcal{X}_A = \{x \in \mathcal{X} : h(x) \leq \ell_0\}$ and $\mathcal{X}_B = \{x \in \mathcal{X} : h(x) \geq \ell\}$, where ℓ_0 and ℓ are two values in $\mathbb{R}$ (usually $\ell_0 = 0$ and $\ell > 0$). Hence, if $h(X(0)) = 0$, κ_A is the first time $\{h(X(t))\}$ down–crosses ℓ_0, whereas κ_B is the first time $\{h(X(t))\}$ up–crosses ℓ.

Splitting [15], [16], [17], [18], [19] is a variance reduction technique for accurately estimating ζ in models where $\{\kappa_B < \kappa_A\}$ is a rare event. In this method, the state space of $\{h(X(t))\}$ is partitioned on the basis of a set of real values $\ell_0 = 0 < \ell_1 < \ell_2 < \cdots < \ell_q = \ell$, as shown in Figure 5(a). Given this partition, for $k \geq 1$, $\kappa_k = \inf\{t > 0 : h(X(t)) = \ell_k > h(X(t^-))\}$ and $\kappa_0 = \inf\{t > 0 : h(X(t)) = \ell_0\}$.

The event $D_k = \{\kappa_k < \kappa_0\}$, $k = 1, 2, \cdots, q$, corresponds to trajectories for which $\{h(X(t))\}$ has up–crossed threshold ℓ_k without entering the region under threshold $\ell_0 = 0$. This definition implies that $D_q \subset D_{q-1} \subset \cdots \subset D_2 \subset D_1$, where $D_q = \{\kappa_q < \kappa_0\} = \{\kappa_B < \kappa_A\}$ is the event of interest,

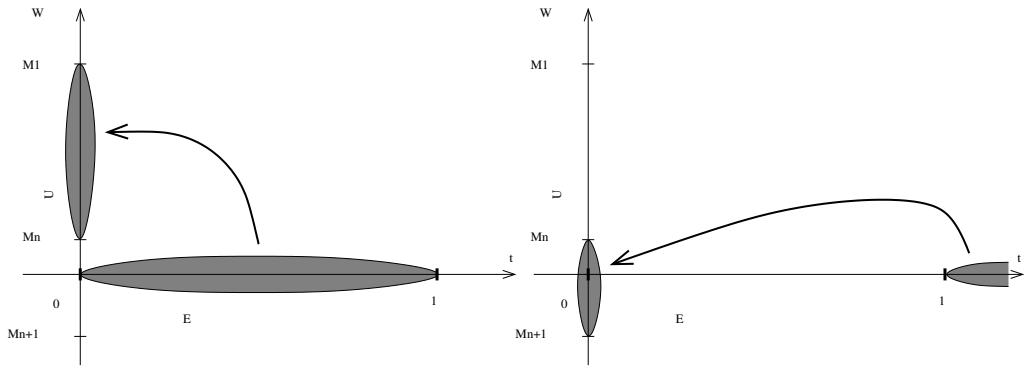

Fig. 3: Transforming Exponential into Uniform distribution

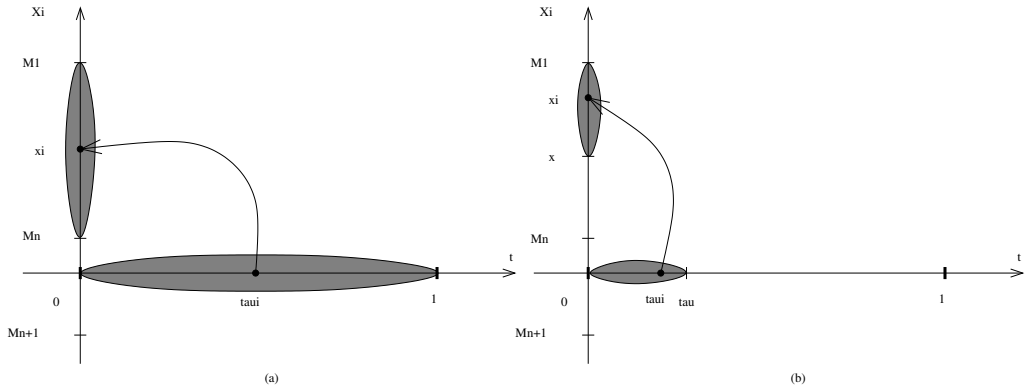

Fig. 4: The function that returns uniformly distributed values, given exponentially distributed values

having probability $\zeta_q = \zeta$. Then,

$$\zeta = \mathbb{P}\{D_q\} = \underbrace{\mathbb{P}\{D_q|D_{q-1}\}}_{p_q} \cdots \underbrace{\mathbb{P}\{D_1\}}_{p_1} = \prod_{k=1}^{q} p_k. \quad (27)$$

The Splitting estimation of ζ is based on this expression. If a set of estimators $\hat{p}_k$ is obtained (as it will be explained next), the estimation of ζ is $\widehat{\zeta} = \prod_{k=1}^{q} \widehat{p}_k$. This estimator has been proved to be unbiased in quite general settings in [20].

At the beginning N_0 replications of $\{h(X(t))\}$ are started at $t = 0$. If R_1 of them up–cross threshold ℓ_1, $\widehat{p}_1 = R_1/N_0$ is an unbiased estimator of p_1. The replications that up-cross ℓ_1 are split (cloned), launching from the crossing points a number $N_1 > R_1$ of new replications towards ℓ_2, each one of them is started with the state of the process at the corresponding crossing point. If R_2 of them reach threshold ℓ_2, $\widehat{p}_2 = R_2/N_1$ is an unbiased (Crude Monte Carlo) estimator of p_2. Proceeding iteratively, the estimators $\widehat{p}_k,\ k = 2, \cdots, q$, are simply $\widehat{p}_k = R_k/N_{k-1}$. If threshold ℓ_q is reached by at least one replication of $\{h(X(t))\}$, the final estimation is: $\widehat{\zeta} = \prod_{k=1}^{q} \widehat{p}_k = (R_1/N_0)\,(R_2/N_1)\,(R_3/N_2) \cdots (R_q/N_{q-1})$.

This mechanism is shown in Figure 5(b). By splitting trajectories, the method privileges the replications for which event $D_q = \{\kappa_q < \kappa_0\}$ is still likely to occur. There are two main implementation variants of Splitting: *Fixed Splitting*, where the number of new copies launched from every hitting point is a constant, $N_k/R_k = \alpha_k > 1\ \forall k$, and *Fixed Effort*, where the number of trajectories started at every hitting point is adjusted so that the total number of replications started from every threshold (*effort*) is a constant, $N_k = F_k\ \forall k$. An important difference to point out is that *Fixed Effort* is not subject to combinatorial explosion on the number of trajectories (which can actually occur in *Fixed Splitting*). This provides a better control of execution times and this is the reason why *Fixed Effort* is the variant selected to support the experimental part of this article.

V. Splitting on the Multi–Level CP

At every single replication of the MLCP, there is one exponentially distributed repair time per link. In order to apply Splitting, one possible sampling plan would be: (i) sample the repair times of all the links, $\{\tau_1, \cdots, \tau_m\}$, (ii) see which of them are less than the time of the first proposed threshold and, (iii) considering that all the links whose repair times are beyond the first threshold are still failed, check the network flow requirements. If the flow values through the network are such that it is necessary to start the second stage of Splitting (comparisons to second threshold), (iv) re–sample multiple copies of all the repair times that in the first stage were beyond the first threshold, but conditioned on the fact that they "are" greater than the first threshold time (due to the lack of memory of the exponential distribution, this consists in sampling the corresponding exponential times and add them to the first threshold time).

Although clear and easy to implement, this sampling plan is inefficient. Many of the repair times sampled in (i) may

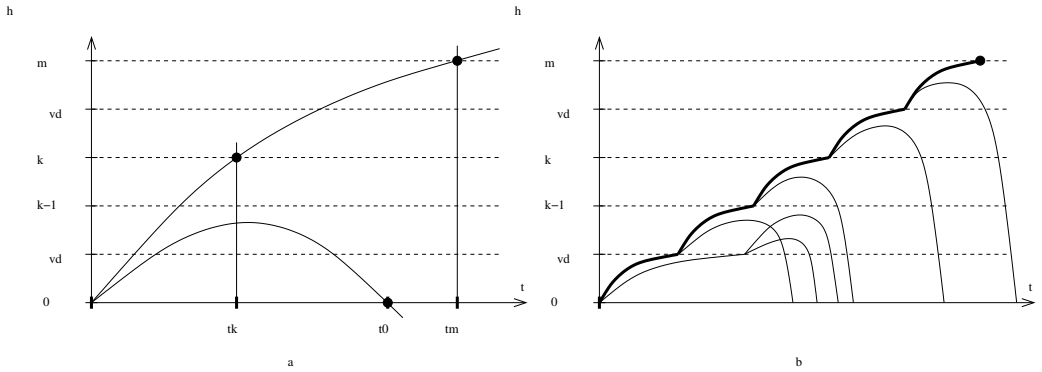

Fig. 5: Sample replications over the state space of $\{h(X(t))\}$

have to be re–sampled in (iv) and possibly over and over in further stages. Besides, in (ii) "all" the times just sampled have to be compared to the time of the first threshold in order to select —probably— a few of them. The sampling plan that we propose, and that is introduced next, solves these inefficiencies by building a trajectory in which the repair times are sampled one at the time, and only once each.

Being $\{\tau_1, \cdots, \tau_m\}$ the set of repair times in the MLCP, call $\Gamma = \{\tau_{(1)}, \cdots, \tau_{(j)}, \cdots, \tau_{(m)}\}$ its *order statistics*, where $\tau_{(1)} \leq \cdots \tau_{(j)} \leq \cdots \tau_{(m)}$. The idea is to see this sequence as a random process subject to the Splitting application. Based on two well known properties of the exponentially distributed r.v., the elements in Γ can be obtained one by one as follows.

Let $\Gamma^i = \{\tau_{(1)}, \tau_{(2)}, \cdots, \tau_{(i)}\},\ 1 \leq i \leq m$, with $\Gamma^0 = \emptyset$, be a partially sampled sequence, and let Λ^i be the set of parameters of the links whose repair times are in Γ^i (already sampled). To build a trajectory it is necessary to obtain Γ^{i+1} given Γ^i, which supposes to find: the time of the next reparation, $\tau_{(i+1)}$, and the link l repaired in time $\tau_{(i+1)}$. Calling $S^i = \sum_{j:\lambda_j \in \Lambda_i} \lambda_j$ and $\bar{S}^i = \sum_{j:\lambda_j \notin \Lambda_i} \lambda_j$, then:

- $\tau_{(i+1)} = \tau_{(i)} + \Delta$, where Δ is sampled from an exponentially distributed r.v. with parameter $\bar{S}^i$.
- The link l is sampled from a discrete distribution in which every unsampled link with parameter λ_j has a probability $\lambda_j / \bar{S}^i$.

Considering that the links are repaired sequentially, one after another, each sequence Γ corresponds to a trajectory of the stochastic process $M(\mathbf{X}(t))$. Two replications of such trajectories are shown in Figures 6(a) and 6(b). In each of them there exists a repair time $\tau_{(c)}$ —the repair time of the *critical link*— such that $\forall t < \tau_{(c)}$ we have $M(\mathbf{X}(t)) < T$, whereas $\forall t \geq \tau_{(c)}$ we have $M(\mathbf{X}(t)) \geq T$.

Event $E = \{M(\mathbf{X}(1)) < T\}$ occurs if the *critical link* is repaired after $t = 1$, meaning that at $t = 1$ (instant when the observation of the network is equivalent to observing the static model) the maximum amount of flow generated in $\mathbf{s}$ that can reach $\mathbf{t}$ is below T. Therefore $\zeta = \mathbb{P}\{E\}$. In a standard simulation of the MLCP, the estimator $\widehat{\zeta}$ is the ratio between the number of successful events E and the total number of replications or, equivalently, the ratio between the number of sequences for which $\tau_{(c)} \geq 1$ and the total number of sampled sequences.

The method proposed in this work is based on interpreting sequences $\Gamma = \{\tau_{(1)}, \tau_{(2)}, \cdots, \tau_{(m)}\}$ as replications of a random process, and applying Splitting to simulate this process. To perform this simulation, the interval $[0, 1]$ is partitioned by a set of thresholds $u_0 = 0 < u_1 < u_2 < \cdots < u_q = 1$. This partition defines $E_k = \{M(\mathbf{X}(u_k)) < T\},\ k = 1, 2, \cdots, q$, as indicator events such that, at $t = u_k$, the maximum flow from $\mathbf{s}$ to $\mathbf{t}$ is below T. In view of the ideas of previous sections, we have $\zeta = \mathbb{P}\{E_q\}$ and $\widehat{\zeta} = \prod_{k=1}^{q} \widehat{p}_k$. The estimators $\widehat{p}_k$ can be obtained separately, according to the following mechanism: start one or more sequences Γ from $t = 0$ and then (i) cancel trajectories where event E_1 does not occur and (ii) split trajectories where event E_1 occurs. This is repeated for all sequences started from u_1, i.e. they are canceled or split at threshold u_2. The procedure is applied until threshold $u_q = 1$ is reached. Finally, $\widehat{p}_k = R_k / N_{k-1},\ k = 1, 2, \cdots, q$, where R_k is the number of sequences crossing threshold u_k and N_{k-1} the total number of sequences actually launched from threshold u_{k-1}. The resulting estimator is $\widehat{\zeta} = (R_1/N_0)\,(R_2/N_1)\,(R_3/N_2) \cdots (R_q/N_{q-1})$.

Figure 9 shows a small example to illustrate the mechanism proposed. At $t = 0$ three trajectories are launched. Only for the one at the top $M(\mathbf{X}(u_1)) < T$, this is the reason why it is the only one to be split, creating three new trajectories towards u_2. Out of these three, only for the one in the middle $M(\mathbf{X}(u_2)) < T$. Then, three new trajectories are launched towards time $u_3 = 1$, out of which only the one at the bottom verifies the condition $M(\mathbf{X}(1)) < T$. Finally, $\widehat{p}_k = 1/3,\ k = 1, 2, 3$, therefore $\widehat{\zeta} = \prod_{k=1}^{3} \widehat{p}_k = 1/27$.

The MLCP fits into the estimation process just as the Creation Process does it in [9], after only a few minimal adjustments on the Splitting stage. The MLCP was designed to be as close as possible to the Creation Process, it is in fact an extension of it. Actually, the Creation Process is a particular case of the MLCP. Both methods generate the values (two in the Creation Process and many —eventually ∞— in the MLCP) following similar temporal patterns.

A said, the Splitting application only requires minimal modifications with respect to the proposal in [9], in which a trajectory is split at any threshold cross if there is no connection between nodes $\mathbf{s}$ and $\mathbf{t}$ which, in terms of flows, means that the maximum possible flow from $\mathbf{s}$ to $\mathbf{t}$ is 0. Here,

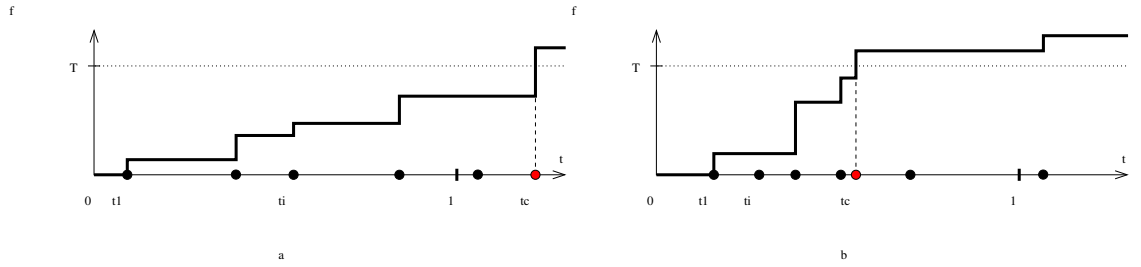

Fig. 6: Two replications $M(\mathbf{X}(t))$ for two different sequences Γ

a trajectory is split at any threshold cross if the maximum possible flow from $\mathbf{s}$ to $\mathbf{t}$ is less that T. It is, therefore, necessary to save a value of flow as part of the saved state, at every threshold cross. Other than these, there are no significant changes with respect to the Splitting application in [9].

Figure 7 shows a flowchart of *Fixed Splitting* on the MLCP. In this example, the ratio between *in* and *out* trajectories per level, i.e. the number by which every trajectory reaching a threshold is multiplied by, is α (the same for all the thresholds). As the mechanism is actually recursive, there is a *stack* that operates as follows. The operation PUSH, performed at time t, saves three values into a single *stack* item:

- the state of $M(\mathbf{X}(t))$,
- the list of the repair times already sampled,
- the value of the next threshold.

The operation POP retrieves the item at the top of the *stack*.

Before starting, three POPs must be executed in order to place on top of the *stack* three identical items containing: $M(\mathbf{X}(0)) = 0$, no repair times already sampled and the value of u_1.

Whenever the value of $M(\mathbf{X}(t))$ is not below T anymore, the trajectory reaches the end (box END in the flowchart) meaning that it will never reach the threshold $u_q = 1$.

If the trajectory is still below T and there are no more repair times to be sampled, the last value obtained for $M(\mathbf{X}(t))$ remains forever, hitting the following threshold, and all the subsequent ones properly multiplied (box HITS in the flowchart). However, in this case, it is not necessary to make the the recursive calls to perform the following thresholds crosses, because it is a fact that the new (split) trajectories will cross all the thresholds ahead. Thus, instead of making the recursive calls, it is more convenient to obtain the number of hits from a simple calculation.

If the threshold just crossed is the last one, only one hit has to be computed (box HIT in the flowchart), but if the threshold is not the last one, then α recursive calls must be made (upwards branch on the left of the flowchart).

After completion of this procedure, the estimation of ζ is computed as $\widehat{\zeta} = (R_1/N_0)\ (R_2/N_1)\ (R_3/N_2)\ \cdots\ (R_q/N_{q-1})$. The values of $R_i,\ i = 1, \ldots, q$ are nothing but the numbers of hits per threshold whereas $N_i = \alpha R_i,\ i = 1, \ldots, q-1$.

Splitting on the MLCP can be easily generalized to the case where the availability and demand are at multiple nodes. In such model, the network should be able to satisfy, simultaneously, the demand of all the pairs source–target. The keys

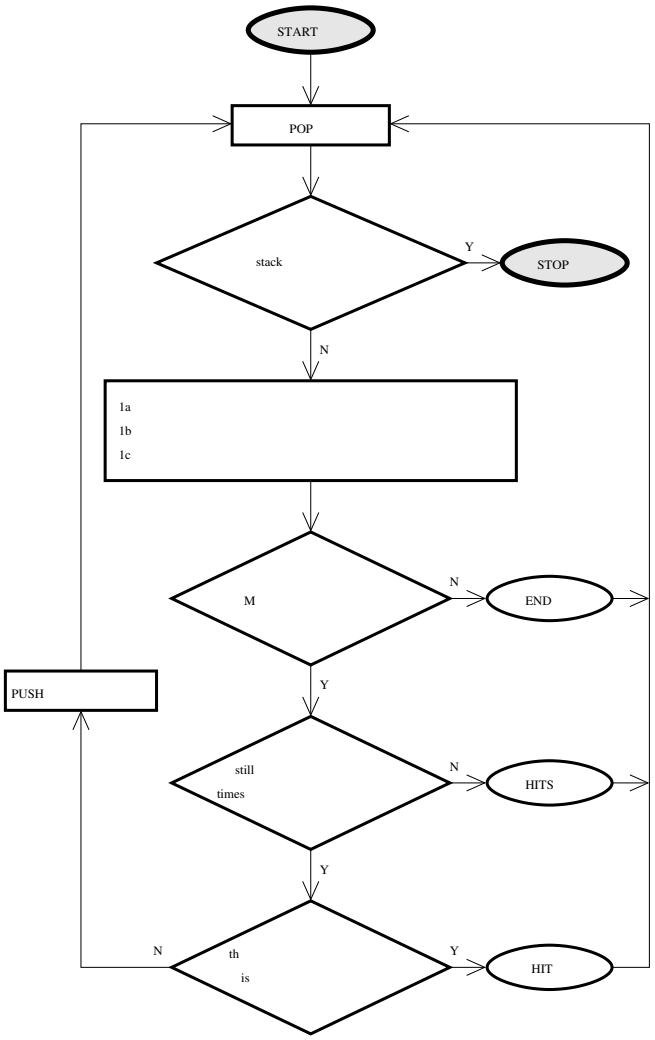

Fig. 7: *Fixed Splitting* on the Multi–Level Creation Process

to implement the adaptation are the following, (i) the main algorithm does not have to change, the successive increments of flow trough all the network, only concerns the links, no matter which and how many are the nodes between which a minimum amount of flow must be guaranteed, (ii) there is not a single trajectory $M(\mathbf{X}(t))$ to observe and to eventually multiply at every threshold cross but as many as pairs source–sink, thus, it is necessary to make multiple comparisons instead of one at every threshold cross and (iii) the adaptation of the

algorithm to compute the maximum possible flow from $\mathbf{s}$ to $\mathbf{t}$ (see the next section) to the case of multiple sources and sinks is straightforward.

VI. Implementation Guidelines

In order to put the preceding ideas into practice, an important issue is how to split trajectories when a threshold is crossed. Let $\Gamma = \{\tau_{(1)}, \cdots, \tau_{(z)}, \tau_{(z+1)}\}$ be a trajectory *under construction*, with $\tau_{(z)} < u_k < \tau_{(z+1)}$ (recall that $\tau_{(z+1)} = \tau_{(z)} + \Delta$, where Δ is a sample of an Exponential random variable with parameter $\bar{S}^{z+1}$). Suppose also that, at time $\tau_{(z+1)}$, $M(\mathbf{X}(t))$ is still below T, meaning that event E_k has actually occurred. Trajectory Γ must therefore be split, in order to create new trajectories, starting from $t = u_k$. To do so, new statistically equivalent values of $\tau_{(z+1)}$ (as many as necessary) have to be be sampled. Due to the properties of the Exponential distribution, these new values can be obtained as $u_k + \Delta$, where Δ denotes a new sampled value, for every new created trajectory.

A problem arising in many Splitting implementations is the considerable computational effort necessary to simulate a trajectories that finally "die". Think, for example, of a model in which, some trajectory started at $\ell_0 = 0$ reaches threshold ℓ_k, but then never reaches the final threshold, ℓ_q. The simulation of such trajectory stops either when it reaches ℓ_{k+1} or — definitely— when it comes back to $\ell_0 = 0$. The fact is that, after leaving threshold ℓ_k, the trajectory might move through the space between ℓ_{k+1} and ℓ_0 for a period of time that that is clearly unavoidable but, from the efficiency point of view, one wish it would be as short as possible. It is to remark that as the distance to the following threshold is, most of the time, less than the distance to threshold ℓ_0, this problem also affects trajectories that actually reach the following threshold but to a much lesser extent. In the model proposed in this article, trajectories grow in only one direction, always toward the next threshold, never coming back. This is why this sort of *up* and *down* trajectories, with all the associated wasted time, is never possible.

The number of thresholds, q, has a direct impact on the efficiency and the accuracy of the Splitting method. Unfortunately there is no efficient procedure to determine the optimal value of q for a general setting. Nevertheless, the analysis of some specific models in the literature, focused on the *basic connectivity network model*, has given rise to some recommendations and guidelines. For instance, in the RESTART variant [21], if the *unreliability* is Q, the optimal value of q is $(\ln Q)/(\ln 0.5) - 1$. Other contributions on this issue, like the ones in [15] and [17], analyze very simple *Fixed Effort* settings, concluding that $q = -(\ln Q)/2$ maximizes the efficiency of the Splitting estimator.

Even when the models used in [21], [15] and [17] do not entirely match the Splitting on the MLCP model, their guidelines for setting q are useful in order to perform an iterative set of pilot runs. As for the first pilot run there is no value of *unreliability* available it is possible to employ, in its place, an upper bound, a lower bound or any value between them. These bounds can be computed using network analysis methods. After the first pilot run, an *unreliability* estimation is available to be used in the next ones.

Every time threshold $u_k,\ k = 1, \ldots, q$, is crossed, it is necessary to check whether condition $M(\mathbf{X}(u_k)) < T$ still holds. In the implementations that support this paper this is achieved by means of the Ford–Fulkerson algorithm [11]. This algorithm perfectly fits the model of Splitting on the MLCP, because it is possible to apply it incrementally, saving a lot of computational time.

The Ford–Fulkerson algorithm is based on an auxiliary model known as *residual graph*: $\mathcal{G}_R(\mathcal{V}, \mathcal{E}, \mathbf{X}_R)$, in which $\mathcal{V}$ and $\mathcal{E}$ are, respectively, the same set of nodes and links as in the original network and $\mathbf{X}_R = (X_{1R}, X_{2R}, \ldots, X_{mR})$ is a set of *residual capacities* —one component per link— which will be defined next.

Let $l_i(v_j, v_k),\ i = 1, \ldots, m$, be a directed link, from node v_j to node v_k. The *residual capacity* of link l_i is $X_{iR} = X_i - Y_i$, where X_i is the capacity of link l_i in the original network and Y_i is the net flow passing through it, from v_j to v_k. The algorithm uses the fact that $Y_i = Y_i^+ - Y_i^-$ where Y_i^+ is the flow passing through l_i, from v_j to v_k, and Y_i^- is the flow passing through l_i from v_k to v_j, conditional on the fact that $Y_i^+ \geq Y_i^-$.

As M is the maximum amount of flow generated in node $\mathbf{s}$ that can reach node $\mathbf{t}$, the standard Ford–Fulkerson algorithm starts considering $M = 0$ and $X_{iR} = X_i,\ i = 1, \ldots, m$. Then, it attempts to find *augmenting paths*, ψ, in $\mathcal{G}_R$. After each new ψ is found M is increased by M_ψ, the maximum possible flow trough ψ. This process continues until no more *augmenting paths* are found in $\mathcal{G}_R$.

The mechanism that we propose is depicted in the flowchart of Figure 8. It takes advantage on the fact that, the *augmenting paths* detected at any instant are still valid in a later computation of M, even though some links are repaired in the mean time. Suppose that M is computed at time t, for which a set of *augmenting paths* is detected. If M is computed again at time $t + \Delta t$, provided that some links have been repaired between t and $t + \Delta t$, the *augmenting paths* detected at time t are still valid and useful. Besides, the staring point for the new computation is the value of M determined at time t.

The algorithm is called at times $t = u_k,\ k = 1, \ldots, q$, i.e. at every threshold cross. It starts by "reading" the value of $M(\mathbf{X}(u_{k-1}))$ determined in the previous application (0, for the first application), the value of $\mathbf{X}_R$ at time $t = u_k$ what is the same as the *augmenting paths* already found and the set of links repaired since $t = u_{k-1}$ (see the first two blocks in the flowchart). When the algorithm finds the value of $M(\mathbf{X}(u_k))$ and, therefore, "stops", this value and the resulting set of *residual capacities* are "saved" in order to be resumed in the next application.

In the model described by Figure 2 (b), the largest *capacity*, M_1, is associated with the lowest repair times, $[0, t_1)$. The second largest *capacity*, M_2, is associated with the next set of repair times, $[t_1, t_2)$. According to this rule, *capacities* decrease as the corresponding repair times increase, which means that largest *capacities* are sampled earlier. But this is definitely not a requirement, Splitting on the MLCP operates as well if the *capacity* assignment follows any other pattern.

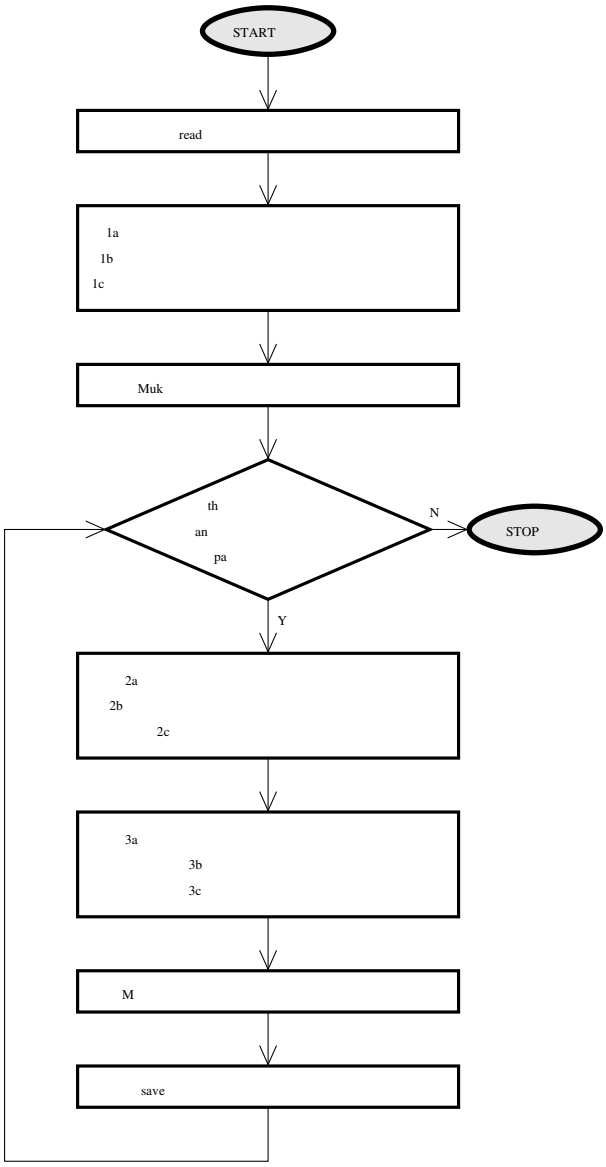

Fig. 8: Incremental application of the Ford–Fulkerson Algorithm

However, as it will be shown next, there is a clear advantage in favor of the variant in which larger *capacities* are sampled earlier.

As networks considered here are highly reliable, most trajectories exceed T earlier than $t = 1$. Call them *unsuccessful* trajectories. After any trajectory is born and until the moment it exceeds T, there is some work to be done and certain amount of splitting to be performed. This work, the amount of splitting and, consequently, the time elapsed, are proportional to the number of thresholds crossed. Thus, if largest *capacities* are sampled earlier, most *unsuccessful* trajectories will reach T "soon", crossing few thresholds and requiring "little" work. If, on the other hand, trajectories reach T closer to $t = 1$, which would be obtained by sampling the smallest *capacities* earlier, there will be more work to be done and more time will be consumed to simulate *unsuccessful* trajectories.

Besides the implementation in [5] —already introduced in Section I—, there is a recent proposal in [22], in which Generalized Splitting is applied to *stochastic flow network's* reliability estimation. The algorithm proposed in [22] has some points in common with the method proposed in this article. It implements, by means of a clever algorithm, an extension of the Creation Process in order to model a network in which links assume multiple values. Unlike the method in this article —in which there is one exponentially distributed time per link— in [22] there is one exponentially distributed time per value that a link can assume. This seems to make impossible the passage to continuous distributions (infinite exponentially distributed random variables per link would be necessary) but, on the other hand, it makes possible the application of Permutation Monte Carlo, which is the most natural and well known variance reduction method associated to the Creation Process. The method proposed in this article can not work with Permutation Monte Carlo but it supports continuous distributions.

Considered over a *stochastic flow network* model, some differences between the classical Splitting, used in this article, and Generalized Splitting —which is a method of very extensive scope— are worth commenting. The sampling mechanism of classical Splitting is a fundamental part of our proposal and was extensively explained in this article. One of its most important issues is the fact that, for each trajectory in stage k, only a "few" repair times have to be sampled (in general, a number much smaller than the number of links). In stage k of Generalized Splitting, in which there are no trajectories, a number N_k values of $\mathbf{X}$ have to be sampled (in response to $R_k < N_k$ successful values of $M(\mathbf{X})$ in stage $k-1$). Here, N_k samples of $\mathbf{X}$ are obtained from a conditional distribution, via Gibbs sampling, after a proceeding that consists in (i) set the capacity to be sampled, temporarily to a provisional value, (ii) compute $M(\mathbf{X})$ considering that provisional value and, (iii) according to that comparison sample an appropriate capacity value. The classical Splitting sampling mechanism seems to be quite simpler (and less time consuming), because conditioning on the success in the previous stage is implicit in the fact that the corresponding threshold is crossed, being necessary no additional computations, nor comparisons.

VII. Experimental Analysis

The experimental phase of this article is based on four network topologies taken from the literature. These networks, shown in Figures 10, 11, 14 and 15, have been widely used for computational studies and benchmarking purposes. The one in Figure 10, called here *Fishman* Network, has been proposed in [6] and used in [5], [7], [6]. The one in Figure 11, known as Dodecahedron, has been widely used in many dependability papers [10], [9] and has been adapted to the *stochastic flow network* problem in [4]. The ones in Figures 14 and 15, called here, respectively, Lattice 4×4 Network and Lattice 6×6 Network, are taken from [22].

In all the experiments $\widehat{\zeta}$ is the estimator of unreliability obtained after simulation time t, with expectation $\mathbb{E}\{\widehat{\zeta}\} = \zeta$ and variance $\mathbb{V}\{\widehat{\zeta}\}$, whereas $\widehat{\zeta}_c$ is the corresponding estimator

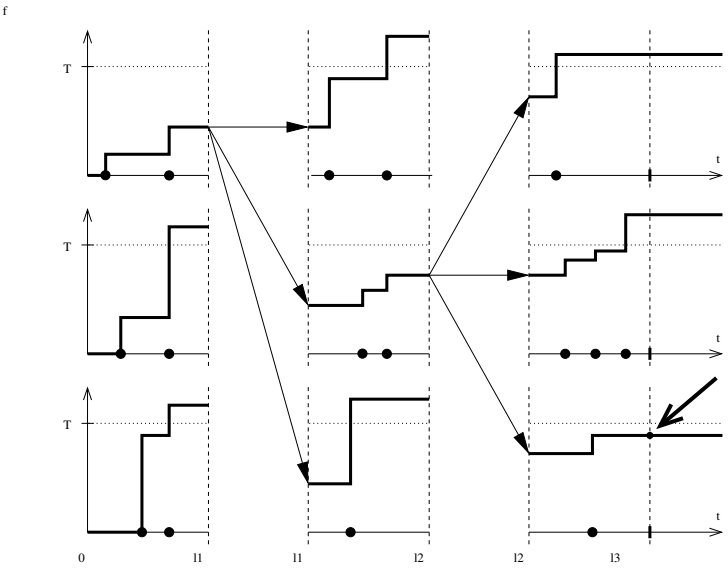

Fig. 9: Splitting on the Multi–Level Creation Process

obtained by Crude or Standard Monte Carlo after simulation time t_c, with expectation $\mathbb{E}\{\widehat{\zeta}_c\} = \zeta$ and variance $\mathbb{V}\{\widehat{\zeta}_c\}$. The following notation applies hereafter:

- $\mathbb{RE} = \mathbb{V}\{\widehat{\zeta}\}^{1/2}/\mathbb{E}\{\widehat{\zeta}\}$, the *Relative Error* ($\mathbb{RE} \times 100$ to indicate it as a percentage).
- $\mathbb{W} = (\mathbb{V}\{\widehat{\zeta}_c\} \times t_c)/(\mathbb{V}\{\widehat{\zeta}\} \times t)$, the *Speedup* or *Precision Gain*, as it shows the precision improvement of the MLCP over Crude Monte Carlo for a fixed computational time. This index can also be interpreted as the computing time improvement for a given precision.

In order to compute $\mathbb{W}$ it is necessary to have a reliable Crude Monte Carlo estimation of ζ, which would require a sample size N much larger than $1/\zeta$. If $N \leq 1/\zeta$, the number of samples for which the network fails tends to 0, and so do the crude estimator $\widehat{\zeta}_c$, its variance $\mathbb{V}\{\widehat{\zeta}_c\}$ and the product $\mathbb{V}\{\widehat{\zeta}_c\} \times t_c$. Nevertheless, the true value of $\mathbb{V}\{\widehat{\zeta}_c\}$ is $\zeta(1-\zeta)/N$ and, therefore, $\mathbb{V}\{\widehat{\zeta}_c\} t_c = \zeta(1-\zeta)/(N/t_c)$. As (N/t_c) is a constant, the product $\mathbb{V}\{\widehat{\zeta}_c\} \times t_c$ is a constant as well. Finally, as long as the true value of ζ (or, at least, a precise estimation) is known, the only necessary value to determine $\mathbb{V}\{\widehat{\zeta}_c\} \times t_c$ is (N/t_c), which can be determined from a not necessarily too long run of the Crude or Standard simulation.

The *Fixed Effort* variant of Splitting on the MLCP was programmed in the `C` language, using the `gcc` compiler. Each experiment comprises K trials, every one performed by launching F trajectories of a *Fixed Effort* simulation. Each of the K trials estimates a value $\widehat{\zeta}_k$. The overall estimation is:

$$\widehat{\zeta} = \frac{1}{K} \sum_{k=1}^{K} \widehat{\zeta}_k. \tag{28}$$

Given that the exact variance of $\widehat{\zeta}$ is unknown, the values of $\mathbb{RE}$ and $\mathbb{W}$ are obtained using $\widehat{\mathbb{V}}$, an unbiased estimator of $\mathbb{V}\{\zeta\}$, computed from the results of the experiments using this formula:

$$\widehat{\mathbb{V}}\{\widehat{\zeta}\} = \frac{1}{K-1} \left[\frac{1}{K} \left(\sum_{k=1}^{K} \widehat{\zeta}_k^2 \right) - \widehat{\zeta}^2 \right]. \tag{29}$$

The sample size of the simulations is $N = K \times F$.

The first set of experiments examine the efficiency of Splitting on the MLCP for continuous models over the topologies considered in this section. In both networks $X_i = 0$ with probability p_0 and, whenever $X_i \neq 0$, $X_i \sim \text{Unif}(100, 200)$, $i = 1, \ldots, m$. In all the experiments $F = 10^3$ and $K = 10^3$. The results are shown in Tables I and II. The number of thresholds selected are pointed out next to each value of $\mathbb{RE}$.

The second set of experiments is quite similar to the first one. It is also based on the *Fishman* Network and the Dodecahedron. The only change concerns the distribution of the link's *capacities* that now is discrete, according to the following pattern: $X_i = 0$ with probability p_0 and X_i equals, respectively, 100 or 200, with probability $(1 - p_0)/2$. Again, in all the experiments $F = 10^3$ and $K = 10^3$. The results are shown in Tables III and IV. The number of thresholds selected are pointed out next to each value of $\mathbb{RE}$.

The third, and last, set of experiments is intended to make a qualitative evaluation of two improvements suggested in Section VI, namely, the incremental application of the Ford–Fulkerson algorithm and the way the *capacity* values per link are arranged in time (largest *capacities* are sampled earlier). To give support to these experiments, the implementation of Splitting on the MLCP was modified. The modifications consist in canceling, alternatively, each of the improvements to be evaluated. In one of these implementations the Ford–Fulkerson algorithm is applied from scratch every time, i.e. its application is not incremental. Using this variant the corresponding estimator is obtained in time t_{NI} (NI means "not incremental"). In the other implementation variant the *capacity* values are arranged the other way around, here the simulation time is t_{SE} (SE means "smallest *capacities* are sampled earlier"). Tables

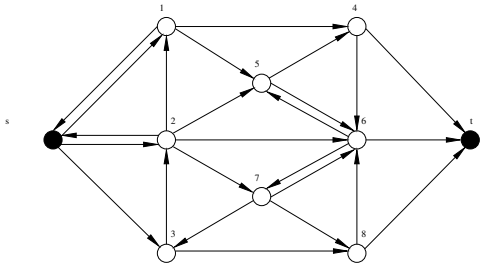

Fig. 10: *Fishman* Network

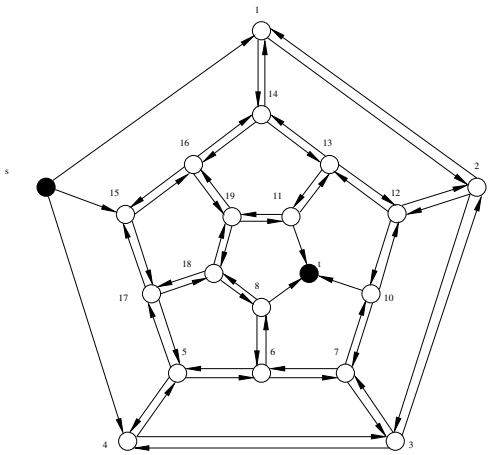

Fig. 11: Dodecahedron

V and VI attempt to illustrate the efficiency of these two improvements. Table V shows the ratio t_{NI}/t, a factor that quantifies the improvement attained by using the incremental version of the Ford–Fulkerson algorithm, whereas Table VI shows the ratio t_{SE}/t, a factor that quantifies the improvement attained by the scheme in which largest *capacities* are sampled earlier.

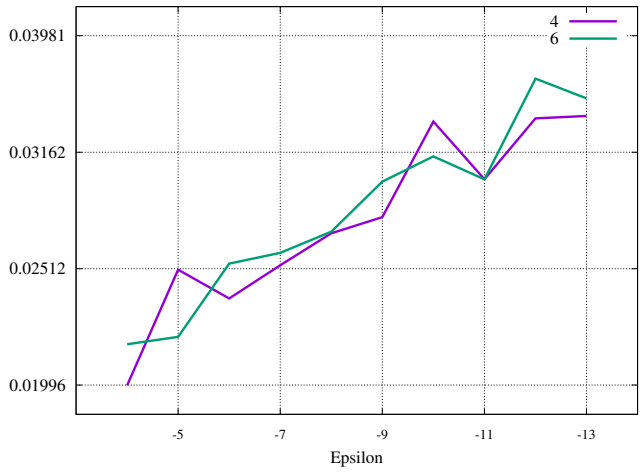

Fig. 12: Relative Error

For all the experiments, the *relative errors* grow as expected in all Splitting–based implementation, the higher reliability the higher *relative error*. See that the reliability grows either if the single links reliability grow or if the value of T decreases, and the *relative errors* do it accordingly. Anyway, *relative errors* under 1% for unreliabilities in the order of 10^{-9} and 10^{-12} —as can be found in all the tables— are not easily reachable for any variance reduction method. See that for these levels of unreliability, and for all the models, the values of the *speedups* are huge, indicating that, in those ranges of values the standard simulation is directly unfeasible. The *speedup* tends to be larger when the network is larger (see Dodecahedron vs. *Fishman* Network, for the same range of reliabilities and the same distributions), because the execution

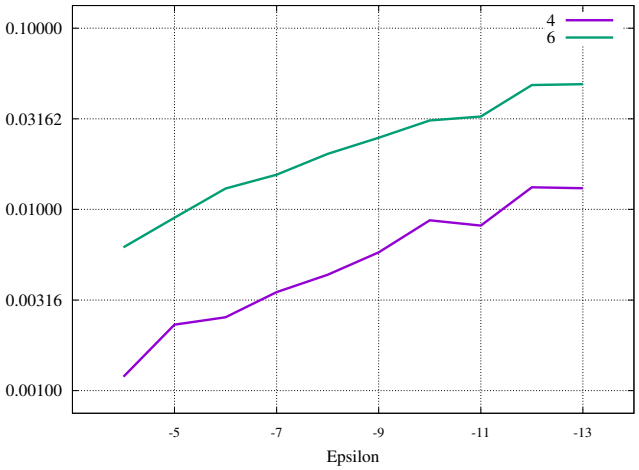

Fig. 13: Work Normalized Relative Variance

time of Crude Monte Carlo grows together with the size of the network, but this is not strict because both, the variance and the execution times, are affected by the number of thresholds, what makes the comparison not so direct. The improvement attained by the incremental application of the Ford Fulkerson algorithm exhibits an evolution that is, apparently, not determined by any parameter of the simulation what, considering that all the values are quite close, might be indicating that the improvement is approximately the same for all the experiments. The improvement attained by sampling earlier the larger *capacities* seems to increase together with the reliability.

The final part of this section provides an experimental comparison. Some of the experiments performed in [22], which are based on the Lattice 4×4 and the Lattice 6×6 network models, are reproduced here using Splitting on the MLCP as the method for the estimations. Four methods have been introduced, and tested, in [22], all of them based on a on a graph evolution model specifically designed to be the basis of Generalized Splitting (one of the methods, referred to as GS) and also the basis of Permutation Monte Carlo (three more methods referred to, respectively, as PMC, PCM–Single

and PMC–All).

In all the experiments the efficiency is assessed by means of the *relative error* $\mathbb{RE}$ (already defined) and the *work–normalized relative variance* defined as $\mathbb{WNRV} = t \times \mathbb{RE}^2$ (the experiments that support this article were run on a linux PC with processor Intel Core 2 Duo T6600). In all cases, the network models are such that $X_i = k,\ k = 0, 1, \ldots, n_i$, $\mathbb{P}\{X_i = k\} = \rho^{n_i-k-1}\varepsilon, k < n_i$ and $\mathbb{P}\{X_i = n_i\} = 1 - \sum_{k=0}^{n_i-1} \rho^{n_i-k-1}\varepsilon$. In order to be as close as possible to the setting of the experiments in [22], where the sample size is 5×10^{-4}, we performed $K = 10^2$ trials, in each one of which $F = 5 \times 10^2$ trajectories were launched. The number of thresholds selected was computed as suggested in [22], that is $q \approx -\ln\zeta$. The parameters selected were: $n_i = 8,\ i = 1, \ldots, m$, $\rho = 0.8$ and $T = 10$, whereas ε ranged between 10^{-4} and 10^{-13}.

Table VII shows a comparison with those results obtained in [22] by Generalized Splitting (GS) and Permutation Monte Carlo (PM), based on the Lattice 4×4 . The values achieved by Splitting on the MLCP —on top of the table— outperform all those corresponding to the other two methods.

Figures 12 and 13 show graphically the same values as those in Table VII for the case of the Lattice 4×4 and also for the Lattice 6×6, but with ε reaching up to 10^{-13}.

Concerning $\mathbb{RE}$ for the Lattice 4×4, the plotted values in Figure 12 are slightly lower than those for PMC–All in [22] when ε is large, and they are approximately the same as those for PMC–All when ε is small (and extremely small). In the case of the Lattice 6×6, the values in Figure 12 are quite similar than those for PMC–All in [22]. See that, from the $\mathbb{RE}$ point of view, PMC–All is the most efficient of the methods introduced in [22].

With respect to the $\mathbb{WNRV}$, which is a parameter that evaluates the temporal performance, almost all the plotted values in Figure 13 are below than the corresponding values in [22]. In fact, the worst value in the Lattice 4×4 graph, which is $10^{-1.9}$, is nearly the same as the best value for the corresponding graph in [22], whereas the worst value in the Lattice 6×6 graph, which is $10^{-1.3}$, is below the best and, as a consequence, below all the corresponding values in [22].

VIII. Conclusions

The main proposal of this article is to present a new *Monte Carlo* method that we have called Splitting on the Multi-Level Creation Process. It is aimed at obtaining accurate *reliability* estimations in the context of highly reliable *stochastic flow networks*. After describing and analyzing the method's operation, its efficiency has been empirically shown.

The Creation Process [10] is a graph evolution model originally conceived to be applied on a *basic connectivity network model*, in which links can be found in one of two possible states. In a *stochastic flow network* links can assume multiple values. This is the reason why the Creation Process was modified and transformed into what we called Multi–Level Creation Process. Once the MLCP was available, the application of Splitting on it completes the method and, therefore, the paper's proposal.

Some methods based on the Creation Process are efficient when networks are not too large, because after each link is repaired it is necessary to apply an algorithm to check the final condition of the network. These algorithms (like Ford–Fulkerson in our proposal) are highly time consuming and the number of its applications is proportional to the size of the network. This is the reason why they consume an important part of the whole simulation time. But if Splitting is applied, the application of these algorithms is only necessary when a threshold is crossed, which supposes that one or more links have been repaired (possibly many). As a consequence, in the method proposed here, the number of times that the Ford–Fulkerson algorithm needs to be applied is significantly lower than the number of applications required by other methods based on the standard Creation Process.

Besides, there are two features that significantly contribute to the efficiency of Splitting on the MLCP, namely, the incremental application of the Ford–Fulkerson algorithm, which has been designed specifically for this work, and the fact of selecting the order in which the links' *capacity* values are assigned, according to their repair times. The efficiency increment due to these improvements has been assessed — and found to be quite important— in Section VII.

Many possible lines of future work can be derived from this article. One of them is to study conditions under which the proposed method can verify the Bounded Relative Error property. Other issues to be studied include the use of analytical methods for determining the number of thresholds and a deeper discussion of accuracy and execution time trade-offs.

Also, another interesting venue of work is to tackle models with dependency among the links' states (which, as works such as [23] explain, can have an important impact in reliability values). In particular, it seems possible to apply the proposed method to cases where the dependence can be represented by a set of exponential random variables, like the case of the Marshall-Olkin copula distributions (arising from shock —and anti–shock— models with exponential time arrivals), which have been studied in the case of traditional network reliability in [24], [25] and [26], and can be extended to multi-level link capacity distributions.

Acknowledgment

This research was partially supported by project MATH–AmSud 19-MATH-03 (RareDep).

We thank the editor and the anonymous reviewers for their useful comments and suggestions which helped us to improve this work.

References

[1] S. Soh and S. Rai, "An efficient cutset approach for evaluating communication-network reliability with heterogeneous link-capacities," *IEEE Transactions on Reliability*, vol. 54, no. 1, pp. 133–144, March 2005.

[2] W. Yeh, "An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability," *IEEE Transactions on Reliability*, vol. 64, no. 4, pp. 1185–1193, Dec 2015.

[3] H. Cancela, L. Murray, and G. Rubino, "Highly reliable stochastic flow network reliability estimation," in *XLII Latin American Computing Conference, CLEI 2016, Valparaíso, Chile, October 10-14, 2016*, 2016, pp. 1–10. [Online]. Available: https://doi.org/10.1109/CLEI.2016.7833343

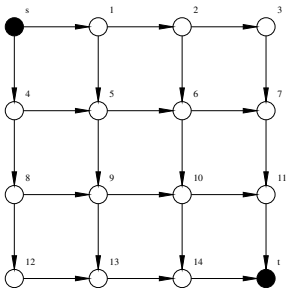

Fig. 14: Lattice 4×4 Network

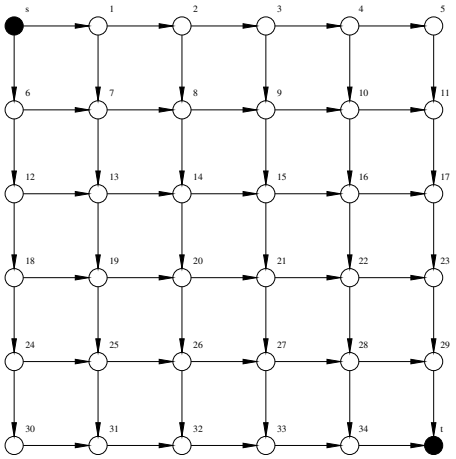

Fig. 15: Lattice 6×6 Network

[4] I. Gertsbakh, R. Rubinstein, Y. Shpungin, and R. Vaisman, "Permutational methods for performance analysis of stochastic flow networks," *Probability in the Engineering and Informational Sciences*, vol. 28, no. 1, pp. 21–38, 2014.

[5] Z. I. Botev, S. Vaisman, R. Y. Rubinstein, and P. L'Ecuyer, "Reliability of stochastic flow networks with continuous link capacities," in *Proceedings of the 2014 Winter Simulation Conference*, ser. WSC '14, Piscataway, NJ, USA, 2014, pp. 543–552. [Online]. Available: http://dl.acm.org/citation.cfm?id=2693848.2693926

[6] G. S. Fishman and T.-Y. D. Shaw, "Evaluating reliability of stochastic flow networks," *Probability in the Engineering and Informational Sciences*, vol. 3, pp. 493–509, 10 1989.

[7] S. Bulteau and M. El Khadiri, "A Monte Carlo simulation of the flow network reliability using importance and stratified sampling," *RAIRO - Operations Research - Recherche Opérationnelle*, vol. 32, no. 3, pp. 271–287, 1998. [Online]. Available: http://eudml.org/doc/105171

[8] P. Doulliez and E. Jamoulle, "Transportation networks with random arc capacities," *RAIRO - Operations Research - Recherche Opérationnelle*, vol. 6, no. V3, pp. 45–59, 1972. [Online]. Available: http://eudml.org/doc/104557

[9] L. Murray, H. Cancela, and G. Rubino, "A Splitting algorithm for network reliability estimation," *IIE Transactions*, vol. 45, no. 2, pp. 177–189, 2013.

[10] T. Elperin, I. B. Gertsbakh, and M. Lomonosov, "Estimation of network reliability using graph evolution models," *IEEE Transactions on Reliability*, vol. 40, no. 5, pp. 572–581, Dec 1991.

[11] D. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[12] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, Apr. 1972. [Online]. Available: http://doi.acm.org/10.1145/321694.321699

[13] A. V. Goldberg and S. Rao, "Beyond the flow decomposition barrier," *J. ACM*, vol. 45, no. 5, pp. 783–797, Sep. 1998. [Online]. Available: http://doi.acm.org/10.1145/290179.290181

[14] J. S. Provan and M. O. Ball, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM J. Comput.*, vol. 12, no. 4, pp. 777–788, 1983. [Online]. Available: https://doi.org/10.1137/0212053

[15] M. J. J. Garvels, "The splitting method in rare event simulation," Ph.D. dissertation, Faculty of mathematical Science, University of Twente, The Netherlands, 2000.

[16] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic, "Splitting for rare event simulation: Analysis of simple cases," in *Proceedings of the 28th conference on Winter simulation, WSC 1996, Coronado, CA, USA, December 8-11, 1996*, 1996, pp. 302–308. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/WSC.1996.873293

[17] P. L'Ecuyer, V. Demers, and B. Tuffin, "Rare events, splitting, and Quasi-Monte Carlo," *ACM Trans. Model. Comput. Simul.*, vol. 17, no. 2, Apr. 2007. [Online]. Available: http://doi.acm.org/10.1145/1225275.1225280

[18] M. Villén-Altamirano and J. Villén-Altamirano, "Restart: A method for accelerating rare events simulations," in *Proceedings of the 13th International Teletraffic Congress*. North–Holland, 1991, pp. 71–76.

[19] P. L'Ecuyer, F. Le Gland, P. Lezaud, and B. Tuffin, "Splitting techniques," in *Rare Event Simulation using Monte Carlo Methods*, G. Rubino and B. Tuffin, Eds. John Wiley & Sons, 2009, ch. 3, pp. 39–61.

[20] M. Amrein and H. R. Künsch, "A variant of importance splitting for rare event estimation: Fixed number of successes," *ACM Trans. Model. Comput. Simul.*, vol. 21, no. 2, pp. 13:1–13:20, Feb. 2011. [Online]. Available: http://doi.acm.org/10.1145/1899396.1899401

[21] M. Villén-Altamirano and J. Villén-Altamirano, "Analysis of restart simulation: Theoretical basis and sensitivity study." *European Transactions on Telecommunications*, pp. 373–385, 2002.

[22] Z. Botev, P. L'Ecuyer, and B. Tuffin, "Reliability Estimation for Networks with Minimal Flow Demand and Random Link Capacities," 2018, pp:1-18. [Online]. Available: https://hal.inria.fr/hal-01745187

[23] Y. Lin, P. Chang, and L. Fiondella, "Quantifying the impact of correlated failures on stochastic flow network reliability," *IEEE Transactions on Reliability*, vol. 61, no. 3, pp. 692–701, Sept 2012.

[24] Z. I. Botev, P. L'Ecuyer, R. Simard, and B. Tuffin, "Static network reliability estimation under the marshall-olkin copula," *ACM Trans. Model. Comput. Simul.*, vol. 26, no. 2, pp. 14:1–14:28, Jan. 2016. [Online]. Available: http://doi.acm.org/10.1145/2775106

[25] O. Matus, J. Barrera, E. Moreno, and G. Rubino, "On the marshall–olkin copula model for network reliability under dependent failures," *IEEE Transactions on Reliability*, pp. 1–11, 2018.

[26] J. Barrera, H. Cancela, and E. Moreno, "Topological optimization of reliable networks under dependent failures," *Operations Research Letters*, vol. 43, no. 2, pp. 132 – 136, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167637714001771

Héctor Cancela holds a PhD. degree in Computer Science from the University of Rennes 1, INRIA Rennes, France (1996), and a Computer Systems Engineer degree from the Universidad de la República, Uruguay (1990). He became a Senior Member of IEEE in 2015.

He is currently Head of the Computer Science Institute at the Engineering School of the Universidad de la República (Uruguay). He was Dean of the Engineering School of the Universidad de la República (2010-2015). He is also a researcher at the National Program for the Development of Basic Sciences (PEDECIBA), Uruguay. His research interests are in Operations Research techniques, especially in network models and stochastic models, applied jointly with optimization methods for solving problems in different areas (communications, transport, biological applications, agricultural applications, etc.). He has published more than 90 full papers in international journals, indexed conference proceedings and book chapters

Dr. Cancela is currently President of CLEI (Centro Latinoamericano de Estudios en Informática), and a former president of ALIO (Asociación Latino Ibero Americana de Investigación Operativa). He is a member of the Uruguayan Engineering Academy (ANIU). He has also participated in the IEEE CS Education Award Selection Committee.

Leslie Murray is an Assistant Professor at the Electronics Engineering School of the Universidad Nacional de Rosario (Argentina), where he obtained the degree of Engineer in Electronics. He received the Master degree in Informatics and the Ph.D. degree in Informatics, both from the Engineering School of the Universidad de la República (Uruguay). His main research interests are in Network Reliability and Monte Carlo Simulation.

Gerardo Rubino received the Ph.D. degree in computer science and the Habilitation degree from the University of Rennes 1 in 1985 and 1995, respectively. He is a Senior Researcher with INRIA (the French National Institute for Research in Computer Science and Control), where he is the leader of the DIONYSOS (Dependability, Interoperability and perfOrmaNce analYsiS of netwOrkS) team. His research interests lie in quantitative analysis of computer and communication systems, mainly using probabilistic models. He also works on the quantitative evaluation of perceptual quality of multimedia communications over the Internet. With Bruno Tuffin, he co-edited the book Rare Event Simulation Using Monte Carlo Methods (Wiley, 2009), and co-authored several of its chapters. He has published more than 200 papers in journals and conference proceedings, in several fields of applied mathematics and computer science, and has performed various editorial tasks and managing activities in research. Dr. Rubino is a member of the IFIP WG 7.3.

TABLE I: Efficiency evaluation in a continuous model.
Fishman Network – $X_i = 0$ w.p. p_0, $X_i \sim \text{Unif}(100, 200)$ w.p. $1 - p_0$, $i = 1, \ldots, m$

Measure	p_0	$T = 300$	$T = 250$	$T = 200$	$T = 150$	$T = 100$
$\widehat{\zeta}$	0.01	3.02E−02	8.12E−03	5.99E−04	3.04E−04	2.03E−06
	0.001	3.01E−03	7.58E−04	5.97E−06	3.00E−06	1.99E−09
	0.0001	3.00E−04	7.51E−05	5.99E−08	3.01E−08	2.00E−12
$\mathbb{RE}$	0.01	0.28% (2)	0.44% (2)	0.44% (5)	0.48% (5)	0.68% (7)
	0.001	0.36% (4)	0.51% (5)	0.60% (7)	0.55% (9)	0.73% (13)
	0.0001	0.44% (5)	0.55% (8)	0.66% (11)	0.61% (14)	0.87% (18)
$\mathbb{W}$	0.01	1	2	21	36	2,219
	0.001	8	13	870	2,009	1,379,390
	0.0001	46	90	57,238	122,686	783,333,333

TABLE II: Efficiency evaluation in a continuous model.
Dodecahedron – $X_i = 0$ w.p. p_0, $X_i \sim \text{Unif}(100, 200)$ w.p. $1 - p_0$, $i = 1, \ldots, m$

Measure	p_0	$T = 300$	$T = 250$	$T = 200$	$T = 150$	$T = 100$
$\widehat{\zeta}$	0.01	3.05E−02	8.28E−03	6.06E−04	3.07E−04	2.05E−06
	0.001	3.01E−03	7.56E−04	6.01E−06	3.03E−06	2.00E−09
	0.0001	0.99E−04	7.48E−05	5.97E−08	2.99E−08	1.97E−12
$\mathbb{RE}$	0.01	0.27% (3)	0.47% (2)	0.44% (6)	0.50% (6)	0.71% (8)
	0.001	0.34% (5)	0.52% (5)	0.58% (8)	0.64% (9)	0.80% (14)
	0.0001	0.41% (6)	0.58% (6)	0.65% (12)	0.71% (13)	0.84% (21)
$\mathbb{W}$	0.01	2	2	22	29	2,187
	0.001	7	13	1,024	1,585	1,245,902
	0.0001	48	95	63,611	99,125	833,356,449

TABLE III: Efficiency evaluation in a discrete model.
Fishman Network – $X_i = 0$ w.p. p_0, $X_i = 100$ w.p. $(1 - p_0)/2$ and $X_i = 200$ w.p. $(1 - p_0)/2$, $i = 1, \ldots, m$

Measure	p_0	$T = 300$	$T = 250$	$T = 200$	$T = 150$	$T = 100$
$\widehat{\zeta}$	0.01	1.55E−02	1.55E−02	3.04E−04	3.04E−04	2.03E−06
	0.001	1.51E−03	1.51E−03	3.00E−06	3.03E−06	2.02E−09
	0.0001	1.50E−04	1.50E−04	3.00E−08	3.01E−08	2.00E−12
$\mathbb{RE}$	0.01	0.31% (4)	0.31% (4)	0.48% (5)	0.48% (5)	0.65% (8)
	0.001	0.43% (4)	0.43% (4)	0.55% (9)	0.53% (11)	0.75% (14)
	0.0001	0.44% (7)	0.44% (7)	0.66% (13)	0.61% (14)	0.87% (18)
$\mathbb{W}$	0.01	3	3	63	63	3,880
	0.001	19	19	4,314	4,101	2,051,563
	0.0001	140	140	184,286	218,140	1,252,427,185

TABLE IV: Efficiency evaluation in a discrete model.
Dodecahedron – $X_i = 0$ w.p. p_0, $X_i = 100$ w.p. $(1 - p_0)/2$ and $X_i = 200$ w.p. $(1 - p_0)/2$, $i = 1, \ldots, m$

Measure	p_0	$T = 300$	$T = 250$	$T = 200$	$T = 150$	$T = 100$
$\widehat{\zeta}$	0.01	1.58E−02	1.58E−02	3.07E−04	3.07E−04	2.06E−06
	0.001	1.50E−03	1.50E−03	3.03E−06	3.01E−06	2.03E−09
	0.0001	1.49E−04	1.49E−04	2.99E−08	2.99E−08	1.97E−12
$\mathbb{RE}$	0.01	0.32% (4)	0.32% (4)	0.50% (6)	0.50% (6)	0.70% (9)
	0.001	0.43% (4)	0.43% (4)	0.62% (9)	0.60% (11)	0.83% (13)
	0.0001	0.51% (5)	0.51% (5)	0.71% (13)	0.71% (13)	0.84% (21)
$\mathbb{W}$	0.01	4	4	64	73	3,822
	0.001	22	21	3,282	3,196	2,103,197
	0.0001	137	141	198,868	186,710	1,461,054,482

TABLE V: Improvement attained by the incremental application of the Ford Fulkerson algorithm. Dodecahedron – $X_i = 0$ w.p. p_0, $X_i = 100$ w.p. $(1 - p_0)/2$ and $X_i = 200$ w.p. $(1 - p_0)/2$, $i = 1, \ldots, m$

Measure	p_0	$T = 300$	$T = 250$	$T = 200$	$T = 150$	$T = 100$
$\widehat{\zeta}$	0.01	1.58E−02	1.58E−02	3.07E−04	3.07E−04	2.06E−06
	0.001	1.58E−03	1.50E−03	3.03E−06	3.01E−06	2.03E−09
	0.0001	1.49E−04	1.49E−04	2.99E−08	2.99E−08	1.97E−12
t_{NI}/t	0.01	1.67	1.72	1.50	1.50	1.29
	0.001	2.08	2.05	1.74	1.85	1.34
	0.0001	2.42	2.48	1.93	1.83	1.39

TABLE VI: Improvement attained by sampling earlier the larger *capacities*. *Fishman* Network – $X_i = 0$ w.p. p_0, $X_i \sim \text{Unif}(100, 200)$ w.p. $1 - p_0$, $i = 1, \ldots, m$

Measure	p_0	$T = 300$	$T = 250$	$T = 200$	$T = 150$	$T = 100$
$\widehat{\zeta}$	0.01	3.02E−02	8.12E−03	5.99E−04	3.04E−04	2.03E−06
	0.001	3.01E−03	7.58E−04	5.97E−06	3.00E−06	1.99E−09
	0.0001	3.00E−04	7.51E−05	5.99E−08	3.01E−08	2.00E−12
t_{SE}/t	0.01	1.10	1.09	1.17	1.21	1.29
	0.001	1.17	1.22	1.14	1.32	1.41
	0.0001	1.26	1.41	1.28	1.41	1.46

TABLE VII: Comparative estimations for the Lattice 4×4 Network

	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$	$\epsilon = 10^{-8}$
$\widehat{\zeta}$	2.96E−05	3.02E−06	3.00E−07	3.01E−08	2.99E−09
$\mathbb{RE}$	1.99E−02	2.51E−02	2.37E−02	2.53E−02	2.69E−02
$\mathbb{WNRV}$	1.98E−03	2.31E−03	2.54E−03	3.50E−03	4.36E−03
$\widehat{\zeta}$ (GS)	2.98E−05	2.99E−06	2.99E−07	3.98E−08	2.99E−09
$\mathbb{RE}$ (GS)	3.43E−02	3.32E−02	3.15E−02	3.30E−02	4.33E−02
$\mathbb{WNRV}$ (GS)	1.07E−02	1.43E−02	1.68E−02	2.35E−02	2.86E−02
$\widehat{\zeta}$ (PMC)	2.99E−05	2.98E−06	2.99E−07	2.99E−08	2.99E−09
$\mathbb{RE}$ (PMC)	3.16E−02	3.34E−02	3.41E−02	3.69E−02	3.74E−02
$\mathbb{WNRV}$ (PMC)	3.17E−02	3.20E−02	3.17E−02	3.62E−02	3.54E−02